Lógica

Unificación

Damiano Zanardini

Graduado/a en Ingeniería Informática
Facultad de Informática
Universidad Politécnica de Madrid
damiano@fi.upm.es

Curso Académico 2010/2011

Introducción

[R65], resumen

Theorem-proving on the computer, using procedures based on the fundamental theorem of Herbrand concerning the first-order predicate calculus, is examined with a view towards improving the efficiency and widening the range of practical applicability of these procedures. A close analysis of the process of substitution (of terms for variables), and the process of truth-functional analysis of the result of such substitutions, reveals that both processes can be combined into a single new process (called resolution), iterating which is vastly more efficient than the older cyclic procedures consisting of substitution stages alternating with truth-functional analysis stages.

The theory of the resolution process is presented in the form of a system of first-order logic with just one inference principle (the resolution principle). The completeness of the system is proved; the simplest proof-procedure based on the system is then the direct implementation of the proof of completeness. However, this procedure is quite inefficient, and the paper concludes with a discussion of several principles (called search principles) which are applicable to the design of efficient proof-procedures employing resolution as the basic logical process.

Introducción

Desde [R65]

- traditionally, a single step in a deduction has been required, for pragmatic and psychological reasons, to be simple enough, broadly speaking, to be apprehended as correct by a human being in a single intellectual act
- from the theoretical point of view, however, an inference principle need only to be *sound* and *effective*
- when the agent carrying out the application of an inference principle is a modern computing machine, [...] more powerful principles [...] become a possibility
- in the system described in this paper, one such inference principle is used. It is called the *resolution principle*, and it is machine-oriented, rather than human-oriented

Introducción

Desde [R65]

- the main advantage of the resolution principle lies in its ability to allow us to avoid one of the major combinatorial obstacles to efficiency which have plagued earlier theorem-proving procedures
  - (citado en el artículo) Gilmore
  - (citado en el artículo) Davis-Putnam
  - resolución básica (tal y como la acabamos de ver)

Unificadores

Definición

Una sustitución α es un *unificador* de dos cláusulas F and G si $F\alpha = G\alpha$ (omitiremos a menudo de qué aplicación se trata, podemos verlo como libre)

- en este caso F y G se dicen *unificables*
- un unificador α de F y G se llama *unificador de máxima generalidad* (*UMG*) sii para cualquier otro unificador β de F y G existe una sustitución γ tal que $\beta = \alpha\gamma$

☞ dos fórmulas unificables tienen sólo un (salvo renombrado) *UMG*

Ejemplo: $F = p(x, f(x, g(y)), z)$ y $G = p(v, f(v, u), a)$

- $\alpha_1 = \{\ x/v,\ u/g(y),\ z/a\ \}$ $\quad$ $\alpha_2 = \{\ x/a,\ v/a,\ y/b,\ u/g(b),\ z/a\ \}$
- $F\alpha_1 = G\alpha_1 = p(v, f(v, g(y)), a)$
- $F\alpha_2 = G\alpha_2 = p(a, f(a, g(b)), a)$
- α_1 y α_2 son ambos unificadores, pero α_1 es el *UMG*:

$$\alpha_2 = \alpha_1\gamma \qquad \text{con} \qquad \gamma = \{v/a, y/b\}$$

Algoritmo de Unificación

Varias versiones

- Robinson. [R65]. 1965
- Chang, Lee. Symbolic Logic and Mechanical Theorem Proving. 1973
  - una generalización de la versión propuesta
- Martelli, Montanari. An Efficient Unification Algorithm. 1982
- Escalade-Imaz, Ghallab. A Practically Efficient and Almost Linear Unification Algorithm. 1988
- Henckel. An Efficient Linear Unification Algorithm. 1997
- Suciu. Yet Another Efficient Unification Algorithm. 2006
- y muchos más...

Esta pequeña lista basta para ver que la *eficiencia* es el problema principal

Algoritmo de Unificación

Calcula el *UMG* de dos átomos F y G con el mismo predicado

$\alpha = \lambda$
mientras $(F\alpha \neq G\alpha)$
  encontrar el símbolo más a la izquierda en $F\alpha$ tal que
    el símbolo correspondente en $G\alpha$ sea diferente
  sean t_F y t_G los términos en $F\alpha$ y $G\alpha$ que empiezan con tales símbolos:
    si (ni t_F ni t_G son variables) o
      (uno de ellos es una variable que aparece en el otro)
    entonces FALLO: F y G no son unificables
    en otro caso si (t_F es una variable) **entonces** $\alpha = \alpha(\{t_F/t_G\})$
    en otro caso si (t_G es una variable) **entonces** $\alpha = \alpha(\{t_G/t_F\})$
α es el *UMG* de F y G

Algoritmo de Unificación

Ejemplo: $F = p(x, x)$ y $G = p(f(a), f(b))$

α	$F\alpha$	$G\alpha$	t_F	t_G
λ	$p(x, x)$	$p(f(a), f(b))$	x	$f(a)$
$\{x/f(a)\}$	$p(f(a), f(a))$	$p(f(a), f(b))$	a	b

FALLO: F y G no son unificables

Ejemplo: $F = p(x, f(y))$ y $G = p(z, x)$

α	$F\alpha$	$G\alpha$	t_F	t_G
λ	$p(x, f(y))$	$p(z, x)$	x	z
$\{x/z\}$	$p(z, f(y))$	$p(z, z)$	$f(y)$	z
$\{x/f(y), z/f(y)\}$	$p(f(y), f(y))$	$p(f(y), f(y))$		

F y G tienen un *UMG*: $\{\ x/f(y),\ z/f(y)\ \}$